Recent Advances in the Mercury Monte Carlo Particle Transport Code

M&C 2013
Sun Valley, Idaho
May 5-9, 2013

P. S. Brantley, S. A. Dawson, M. S. McKinley,
M. J. O'Brien, D. E. Stevens, B. R. Beck,
E. D. Jurgenson, C. A. Ebbers, J. M. Hall

Lawrence Livermore National Laboratory

LLNL-PRES-635934
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Mercury is the next-generation general purpose Monte Carlo particle transport code under development at LLNL

- Transports neutrons, photons, and light element (hydrogen and helium) charged particles
- Treats fixed source and criticality problems
- Parallelized via domain replication and domain decomposition
- Written in C++ with a python user interface
- Runs efficiently on current generation massively parallel computing platforms

This talk overviews recent physics and computational science advances in Mercury

- Physics
  - Nuclear resonance fluorescence capability
  - Probability of initiation capability based on probability of extinction

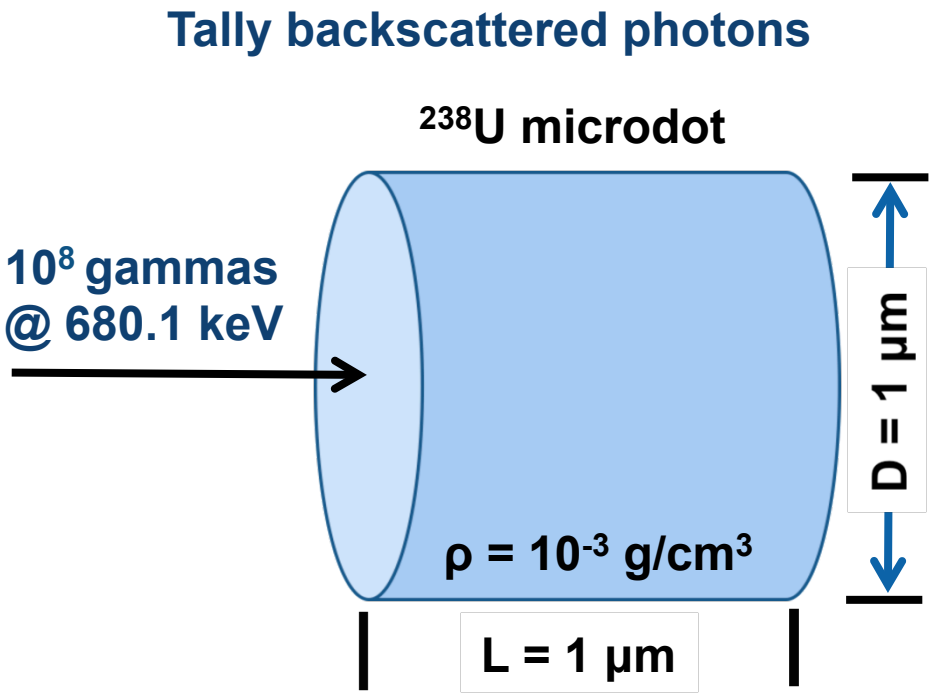
- Computational Science
  - Parallel scaling of algorithms to millions of MPI processes
  - Threading capability

We are improving the physics capabilities of Mercury while enabling the code to run on emerging state-of-the-art computer platforms

An effort at LLNL is developing gamma-ray beams whose energies are tunable and nearly mono-energetic

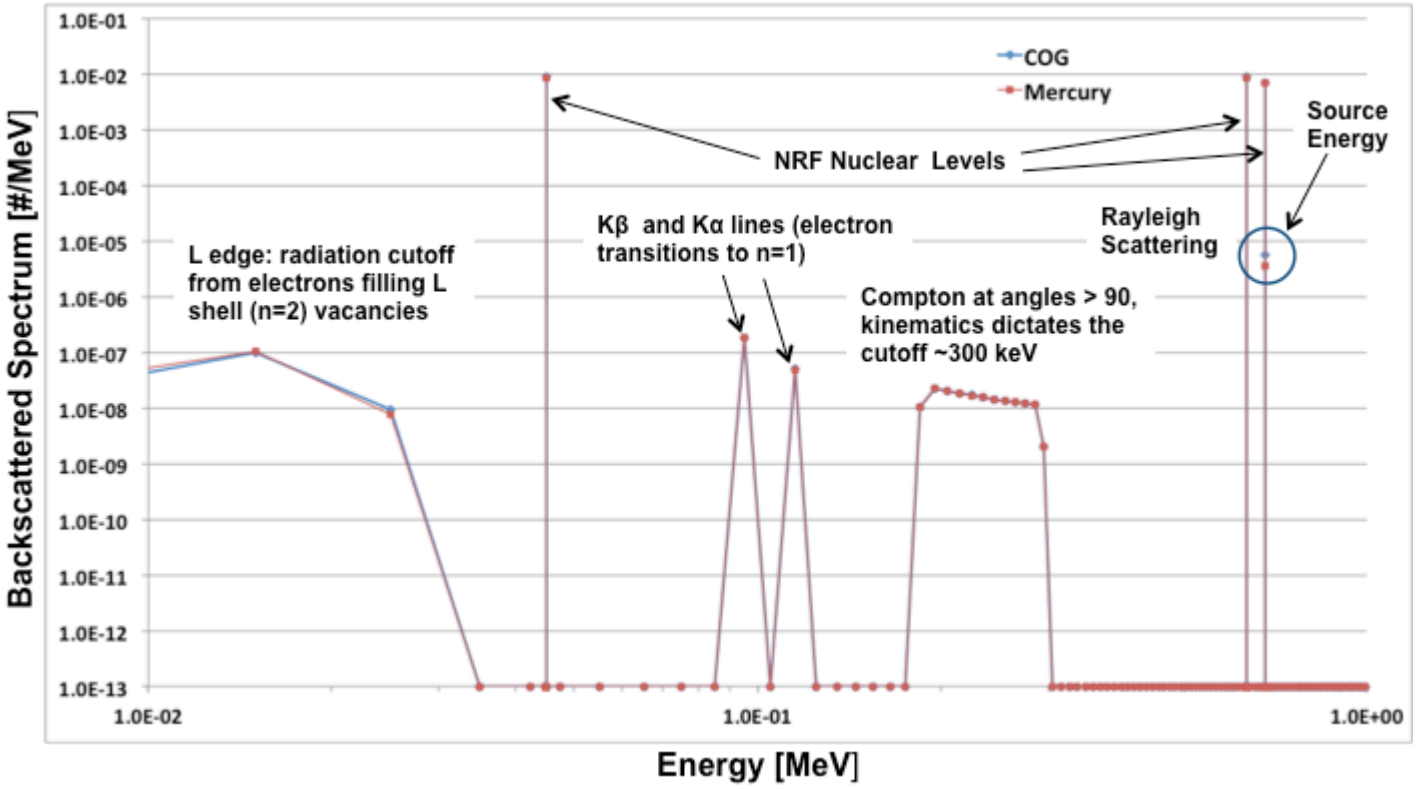
- MEGa-rays (Mono-Energetic Gamma-rays) obtained by Compton upscattering laser photons from high-energy electrons
- Gamma-ray energy can be tuned to the known nuclear resonance fluorescence (NRF) energies of a specific isotope
  - NRF: photon absorbed by nucleus that subsequently decays to ground state through emission of one or more gamma rays with energies characteristic of the isotope
  - → Use to interrogate samples for the presence of an isotope
- Capability to model NRF reactions being added to Mercury to enable its use in ongoing detector design efforts alongside the COG code
  - NRF reaction in addition to Rayleigh (Coherent) and Compton (incoherent) scattering, photoelectric absorption, and pair production
- The MCNPX code also has an NRF capability

We compared the modeling of photon reactions in Mercury and COG for a ^{238}U microdot test problem



- Incident photon energy is at NRF energy for ^{238}U
- Rayleigh, Compton, photoelectric, and NRF reactions are modeled
  - Source photons below threshold energy for pair production
- Goal of test problem is to tally energy spectrum of photons backscattered from microdot
- Modeled in Mercury and COG using 10^8 MC particles

Mercury and COG backscattered spectra are generally in excellent agreement

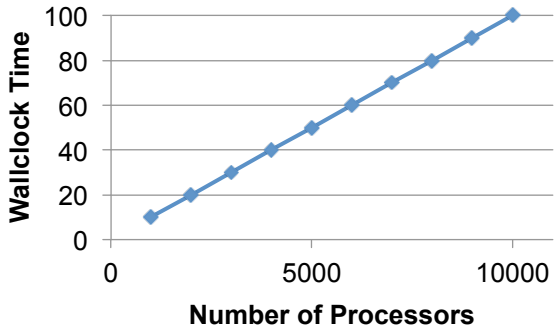


- NRF emission lines in excellent agreement
- Discrepancy in Rayleigh scattering due to more accurate form factor treatment in COG

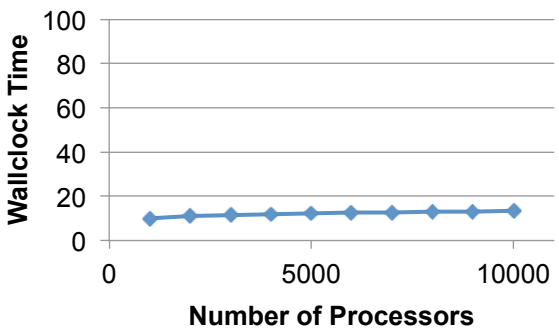
We are improving Mercury to scale to large numbers (i.e. millions) of parallel processes

- Scalability: ability of a code to perform efficiently as the number of parallel processes increases
- Focusing on weak scaling ➔ constant work per process
  - Examines parallel overhead and bottlenecks in the code

Non-Scalable Example: Run time proportional to the number of processors



Scalable Example: Run time proportional to the logarithm of the number of processors



Enabling scalability to millions of processes requires attention to the details of algorithms and memory usage that may be safely ignored at smaller scales

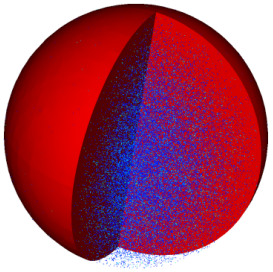
The parallel scalability of several Mercury algorithms has recently been improved

Algorithm	Previous Scaling	Improved Scaling
Particle Sourcing	$O(N_{proc})$	$O(1)$
Global Particle Find – Cartesian Domains	$O(N_{proc}^2)$	$O((\log N_{proc})(\log \log N_{proc}))$
Load Balancing – Replication	$O(N_{proc}^2)$	$O(\log N_{proc})$
Test for Done with Particle Communication	$O(N_{proc}^2)$	$O(\log N_{proc})$

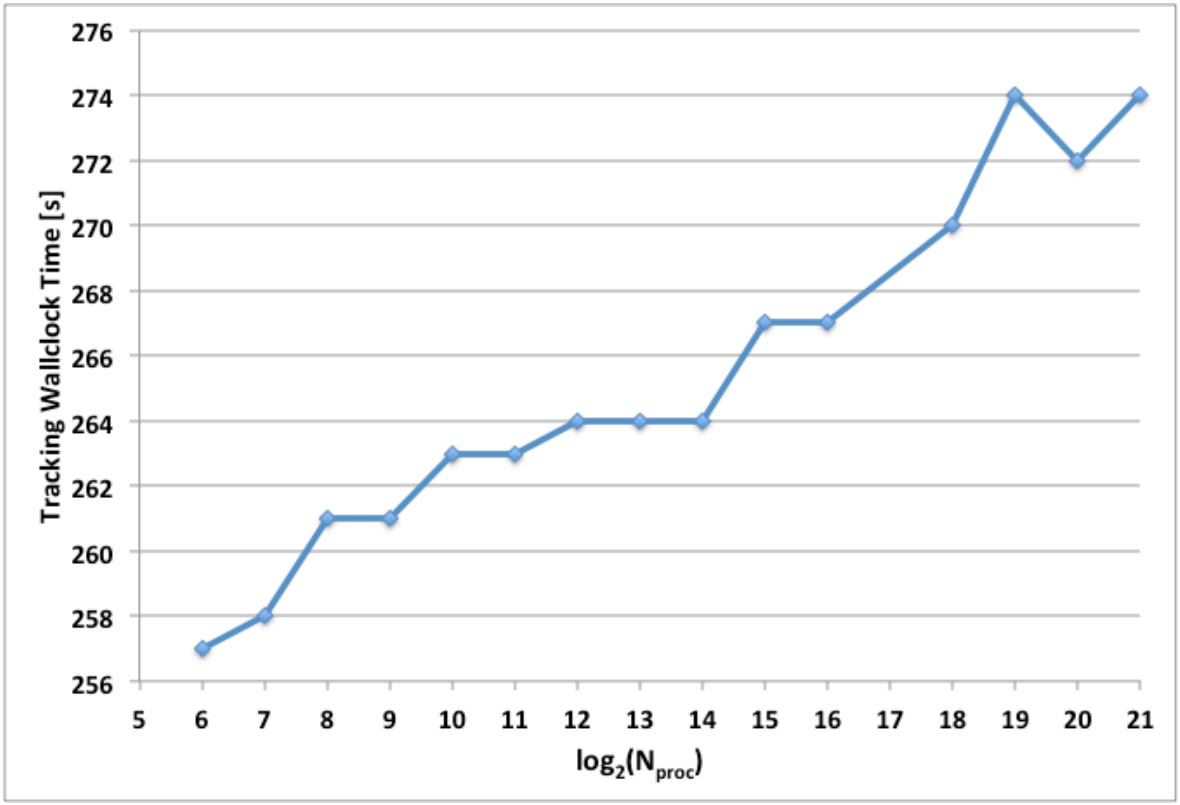
- Sourcing only process-local particles coupled with scalable global particle find algorithm leads to scalable particle sourcing
- Scalable load balancing algorithm considers processor-pairs instead of global workload view
  - See paper/poster by M. J. O'Brien at Monday evening session

We performed a weak scaling study using the Godiva critical sphere benchmark problem

- HEU-MET-FAST-001: highly-enriched uranium sphere, R = 8.7407 cm, ρ = 18.740 g/cm^3
- Weak scaling study performed on LLNL Sequoia supercomputer
  - 20-petaFLOP/s IBM computer with 16 PPC A2 CPU cores per compute node, 4 hardware threads per core, and 16 GB memory per node
- Weak scaling study
  - Pure MPI parallelism, 10^4 MC particles per process
  - $2^6 = 64$ to $2^{21} = 2{,}097{,}152$ MPI processes
  - → Up to ~21 billion Monte Carlo particles tracked for the 2^{21} process case
- Study uses domain replication and tests particle sourcing and load balancing algorithms



Mercury particle tracking wallclock time scales linearly with the $\log_2$ of the number of processors for Godiva



- Tracking time varies by less than 7% when scaling from 64 to 2,097,152 processors
  - Would ideally like constant weak scaling tracking time
- Work is ongoing to improve scalability of additional algorithms in the code

Results of initial scaling study demonstrate parallel scalability to very large numbers of MPI processes

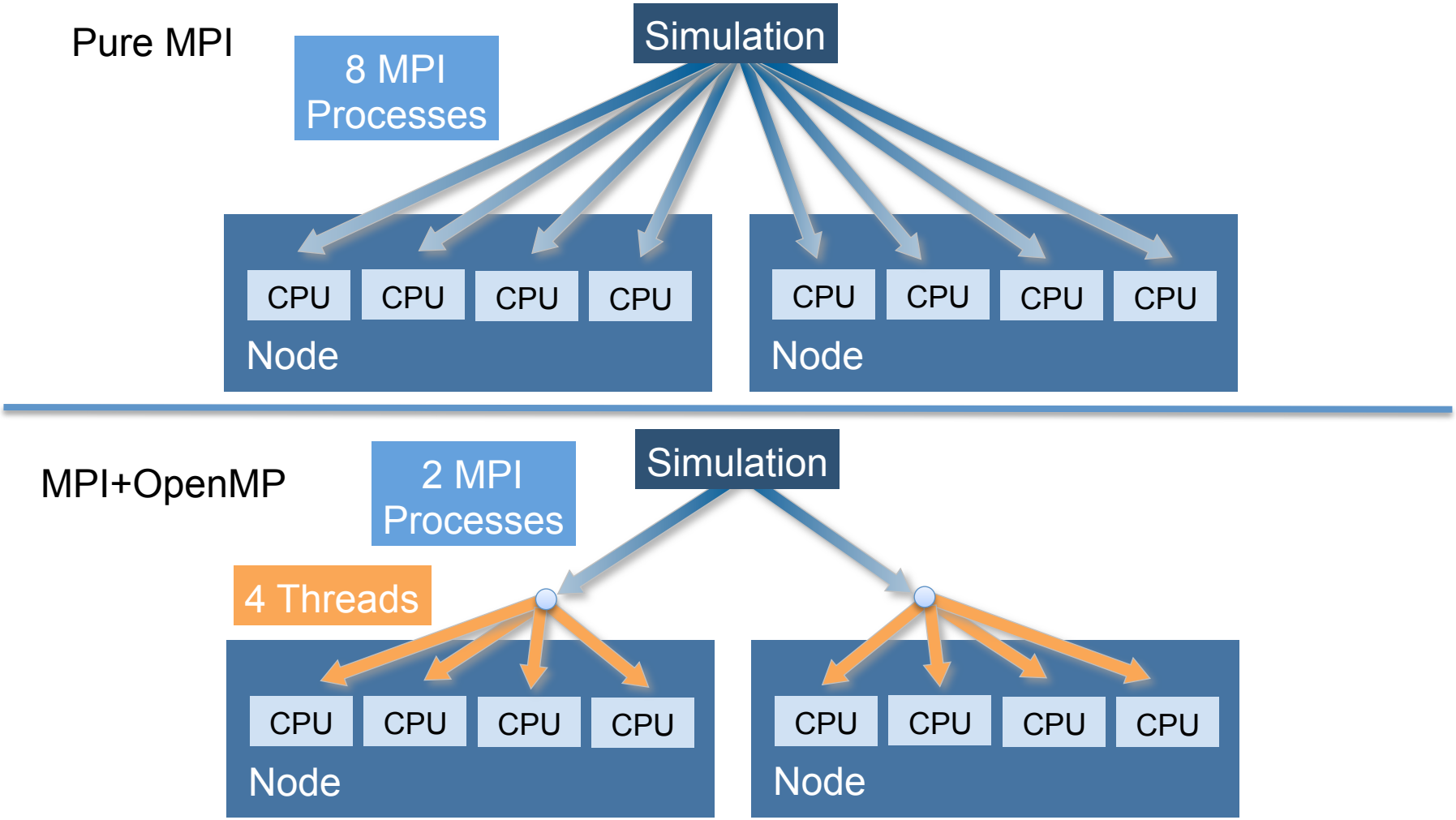
We also implemented a scalable test for when particle communication has been completed

- Algorithm required for domain-decomposed simulations
- Previous algorithm stored $N_{proc} \times N_{proc}$ matrix of number of sent and received messages → algorithm $O(N_{proc}^2)$ and hence not scalable
- Scalable algorithm based on Brunner, Brantley, "An efficient, robust, domain-decomposed algorithm for particle Monte Carlo," JCP (2009)
  - Algorithm is $O(\log(N_{proc}))$: use non-blocking reduce and broadcast operations concurrently as particles are tracking
- Previous published results demonstrated good parallel scaling to ~8K processors on infinite medium weak scaling problems
- We have observed good parallel scaling on Sequoia to 2^{18} = 262,144 MPI processes on infinite medium weak scaling problems
  - Simulations up to 2^{21} = 2,097,152 MPI processes have also been completed → observed some noise above 2^{18} processes

The ability to use OpenMP threads has recently been added to Mercury

- Mercury has traditionally used pure MPI to achieve parallelism
  - MPI used across compute nodes + on CPU cores of individual nodes
- OpenMP threads enable parallelism across the cores of each node
- Simulations can now use a combination of MPI+OpenMP to distribute work across the cores of each node
- Each node may have one or more MPI processes, and each MPI process may use one or more threads to access compute cores
- Memory savings: Nuclear data stored once per MPI process instead of once per CPU core
  - Expected to be important for new machines with larger number of cores per node and lower memory available per node

An MPI+OpenMP simulation uses the CPU cores on a node differently than a pure MPI simulation

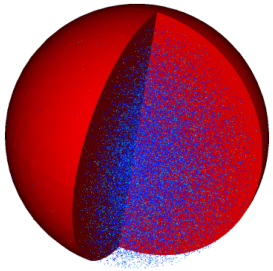


The OpenMP threading capability works with Mercury domain replication and domain decomposition

- Domain replication and domain decomposition distribute work across MPI processes
- OpenMP threading distributes the work of a single MPI process across multiple CPU cores within a node
- MCNP has MPI+OpenMP parallelism for the replication-only case
- Coarse grain threading is achieved by creating a particle *vault* (list of particles to be tracked) for each thread and distributing particles evenly across vaults
  - A thread layer is required in many data structures to enable multiple threads to operate independently without thread locks
- Fine grain threading is also used in lower loop levels outside of the particle processing loop

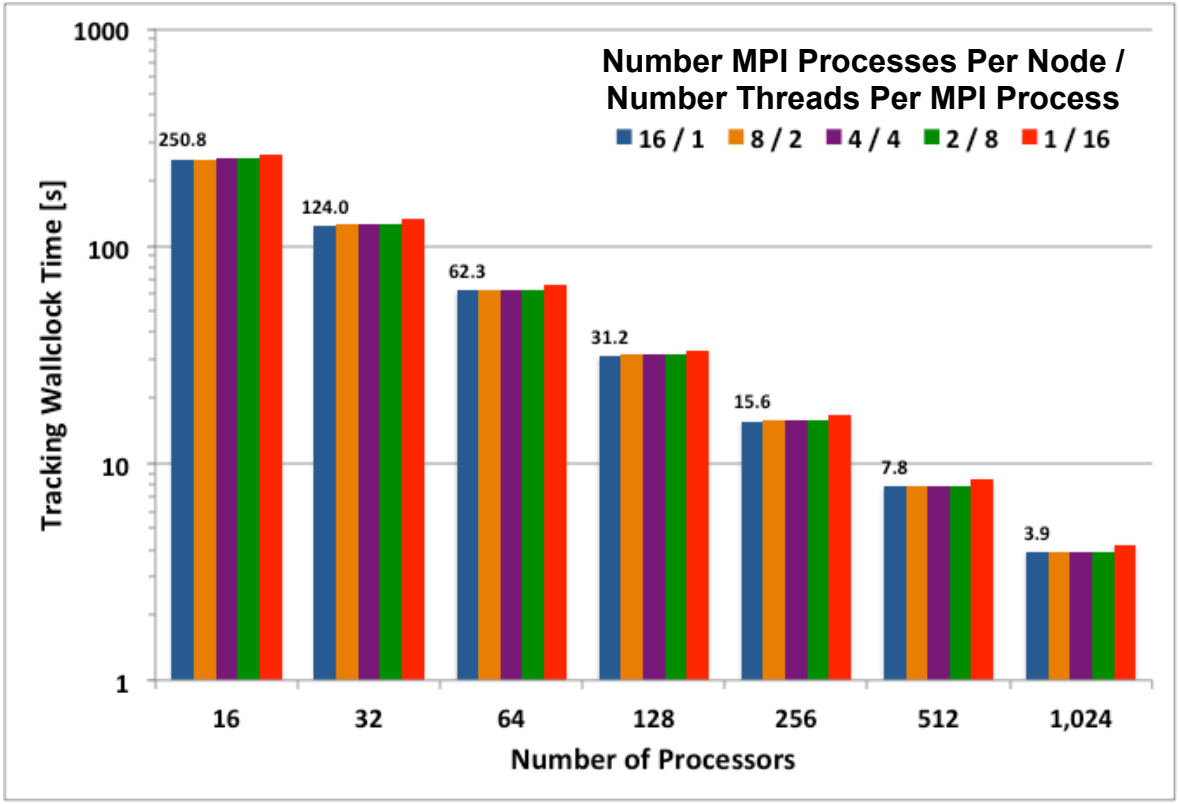
We performed a strong scaling study using the Godiva critical sphere benchmark problem

- Strong scaling: fixed problem size (geometry and number of Monte Carlo particles) and varied the number of parallel processes
- Godiva problem described previously
  - Continuous energy LLNL ENDL2009 nuclear data
  - 10^7 MC particles per generation, k-eigenvalue fractional convergence tolerance 2.5×10^{-4}
  - Simulations give k_{eff} = 1.000188±0.00016 which agrees with experimental value of 1.000±0.001 to within statistics



- Study performed on LLNL RZMerl Linux cluster with 16 Intel Xeon (Sandy Bridge) cores (2.6 GHz) and 32 GB of memory per node
- Investigated particle tracking time and maximum node memory as a function of number of threads per node used (varied from 2 to 16)

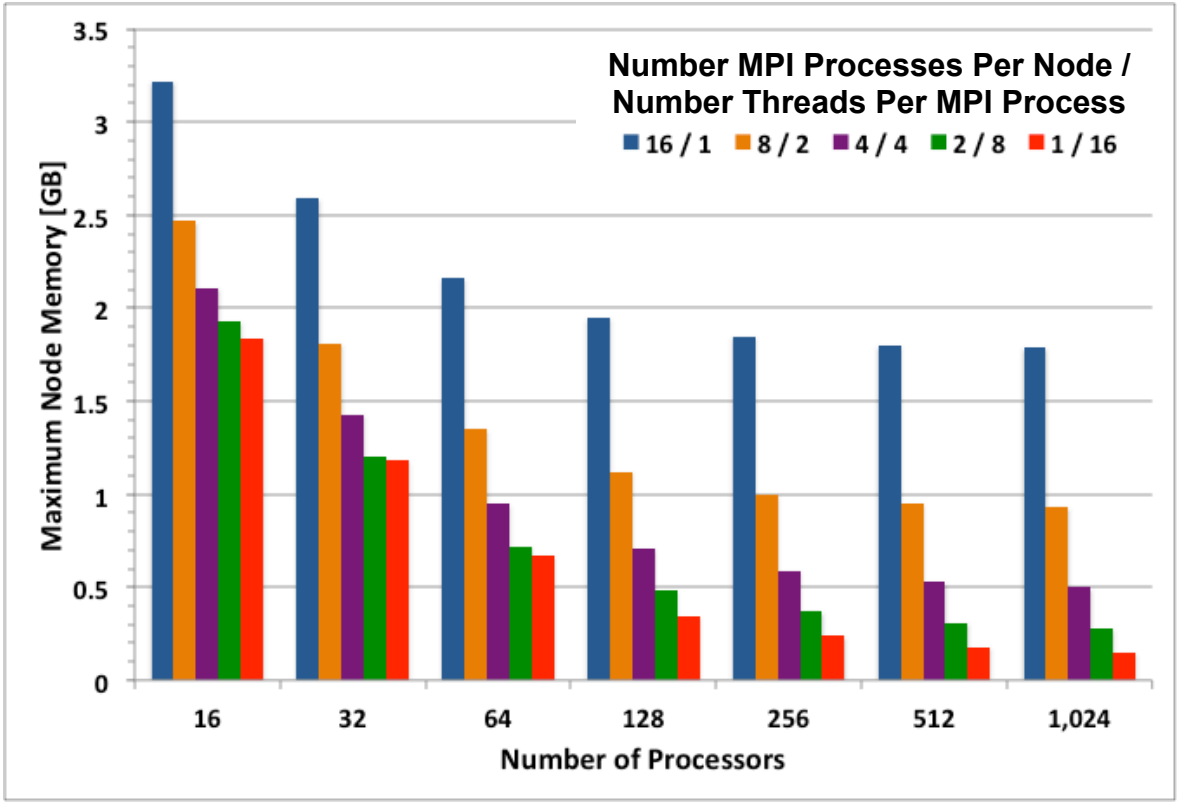
The efficiency of the threading implementation is generally similar to MPI



- Strong scaling excellent – doubling processors reduces tracking time in half
- Tracking time generally insensitive to number of threads (to within ~1-2%)
- Using 16 threads per node (all cores on node) does degrade efficiency
- More opportunities for threading exist in initialization/finalization

Threading implementation for particle tracking is essentially as efficient as using MPI processes within a node

The use of threads for parallelism can significantly reduce memory usage



- Strong scaling with fixed total number of particles → maximum node memory decreases as the number of processors increases
- Nuclear data needs only be stored once per MPI process
- Memory reduction per MPI process elimination ~90-100 MB

We anticipate that the memory reduction achieved by using threads will be important for new machines with less memory per node

We have described recent physics and computational science advances in the Mercury code

- Physics
  - Addition of initial NRF capability is aimed at enabling the application of the code to ongoing detector design efforts

- Computational Science
  - Goal: Enable Mercury to run efficiently on emerging state-of-the-art computers with large numbers of processors and low memory per processor
  - Significant improvements in the parallel scalability of various algorithms
  - Enabling an OpenMP threading option


Lawrence Livermore
National Laboratory